ROBUST CONTINGENCY PLANNING AND SYSTEM DESIGN FOR SAFE AND SECURE AUTONOMOUS ROAD VEHICLES

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Joseph William Corbett-Davies

December 2017

ABSTRACT

Before autonomous vehicles are able to be widely deployed, a number of security and algorithmic challenges must be addressed. Current autonomous vehicles that provide motion safety guarantees exhibit excessively conservative driving behavior when operating in road environments containing highly dynamic obstacles. In this thesis we present a contingency-based motion planning framework for autonomous road vehicles. Probabilistic state predictions are generated for each discrete action of nearby obstacle vehicles, and multiple contingency trajectories are planned such that safe execution is possible under each possible discrete action. An online estimation algorithm is used to infer the discrete obstacle action from sensor observations and inform execution-time contingency selection. We present a fast upper bound on a metric of *distinguishability* that approximates the predicted probability of correctly identifying the discrete action of an obstacle from a set of possible hypotheses. The metric is used to optimize expected execution cost and safety of a set of contingency trajectories. Simulated experiments show that the proposed planning framework produces trajectories with a lower cost and stronger safety guarantees than that of prior work, and this performance improvement persists across a range of vehicle and obstacle initial conditions.

Additionally, a prototype system architecture for a verifiably secure autonomous vehicle is presented. The system architecture is designed to enforce separation of trusted and untrusted information flows. A map verification algorithm is used to verify external data coming from an untrusted source. Motion planning and map verification software components are developed with exist-

ing tools that enforce *information flow control* at the language level. The architecture is implemented on a mobile robotic testbed and experiments are performed to simulate a remote attack scenario. Experimental results show that the architecture is resistant to malicious external data, and can operate safely even when external communications are compromised. Analogies are drawn between the prototype architecture and hardware and software components on real-world autonomous vehicles.

BIOGRAPHICAL SKETCH

Joseph Corbett-Davies completed his undergraduate education at the University of Canterbury in Christchurch, New Zealand, receiving a Bachelor of Engineering (Mechatronics) degree with First-class Honours in 2015. He is currently pursuing a Master of Science degree in the Sibley School of Mechanical and Aerospace Engineering at Cornell University in Ithaca, New York. Joseph is a member of the Autonomous Systems Lab, and his research involves robust decision making, motion planning, and system architecture for autonomous road vehicles. Joseph's graduate study is supported by a Fulbright Graduate Student Award and the J R Templin Trust Scholarship. He has held positions as a Summer Research Scholar at the University of Canterbury (2013, 2014), a teaching assistant at both the University of Canterbury (2013-2014) and Cornell University (2016), and an intern with Autopilot Software at Tesla (May-August 2017).

To Mum, Dad, and Grace.

ACKNOWLEDGEMENTS

Firstly I would like to thank Mark Campbell for his support and dedication during my graduate studies.

The work in Chapter 3 was done alongside Ed Suh, Andrew Myers, Jed Liu, and Rahgava Kumar, who provided invaluable assistance and great feedback. Thank you to Ed in particular, for serving on my committee.

Many classmates and labmates have made the journey an enjoyable and educational one, particularly Reece, Taylor, Yichen, Jen, and Jordan. I would also like to send my thanks to the hard workers at the Farm House: Kyle, Julian, Matthew, Gabriel, Jacob, and Birk. They certainly kept things interesting.

I am grateful for funding support from a Fulbright Graduate Student Award and the J. R. Templin Trust Scholarship.

Lastly, I would like to acknowledge my New Zealand-based support—particularly friends, family, and my talented professors and classmates at the University of Canterbury—without whom I would not have learned much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Motor vehicles kill over a million people per year and are the leading cause of death worldwide for people aged 18 to 29 years [38]. During 2013, in the United States alone, over 32 000 people lost their lives, at a rate more than twice the average for high-income countries, and 2 million were injured due to motor vehicle crashes. This trend does not appear to be universally improving: preliminary data suggests that over 40 000 road users are estimated to have died during 2016, a 14 % increase in deaths since 2014, despite improved vehicle safety technology [37]. In an era of cheap petroleum, automobile-dependent lifestyles, and increasing vehicle kilometers traveled, it is clear that huge steps remain toward providing safety for motor vehicle occupants and, crucially, the cyclists, pedestrians, and other vulnerable road users who make up almost half of road deaths worldwide [38].

While there are proven measures that can mitigate the effect of driver mistakes, such as improved vehicle safety design and the development of safer, slower road infrastructure, the human causes of road crashes—often speed, distraction, or impairment—show no signs of abating. There is a clear argument for reducing and replacing human involvement in driving, both for safety reasons and to extend the convenience of automobile transportation to the young, old, and those with disabilities.

Currently-available vehicles with driver assistance features, such as basic lane-following and automatic braking functionality, have been shown to reduce highway crashes by as much as 40% as compared to vehicles without such features [3]. However, for fully autonomous driving—or even significant driver

assistance—in dynamic urban areas, large advances are required in how automated vehicles are able to perceive and reason about their environments. Nearly 9 years on from initial demonstrations of self-driving technology at the DARPA Urban Challenge [10], there has not been widespread deployment of any vehicles that allow fully autonomous driving without human supervision.

In addition to challenges regarding algorithmic robustness, as autonomous or semi-autonomous vehicles become more widespread they will emerge as a significant security liability and a lucrative target for attackers. Given historical rates of software and device security breaches, it is almost certain that autonomous vehicles developed with existing design techniques and tools will be subject to many attempts—some successful—to maliciously compromise onboard control systems.

Current decision making systems for autonomous cars use overly conservative models of their surroundings to govern vehicle movement. They generally rely on simplistic predictions of the motion of nearby obstacles, and even when sophisticated prediction algorithms are used, the decision making software is often unable to take advantage of the inherent uncertainty in a well-calibrated prediction. Simple models of the environment are adequate, and can even generate safer behavior than a human driver for well structured scenarios, such as nominal highway driving. But as autonomous vehicles begin to operate in the varied and unpredictable environments that define everyday driving, a more considered approach is needed.

This thesis considers two solutions to the above problems. The first is an general motion planning framework for autonomous vehicles that uses contingency plans to enable safe and efficient driving in the presence of obstacles with

uncertain intentions. This work considers the relationship between predictive models of nearby obstacles and the process of identifying obstacle actions from future observations, and uses both of these considerations to inform the trajectory planning process. The proposed method is applied to a simple intersection scenario. Even though prior work can provide trajectory-level safety guarantees at planning time, without consideration of prediction structure these safety guarantees are not valid during trajectory execution. This work is covered in Chapter 2.

In Chapter 3, a system architecture and experimental results are presented for a secure autonomous vehicle platform. A mobile robotic testbed is developed, designed to enforce separation between hardware and software components of differing trust levels, such as those present in a real-world autonomous car. In particular, a path planner is developed using language-level *information flow control* to allow the secure use of both trusted and untrusted information after explicit verification and endorsement. Experiments are conducted to simulate an attack via a compromised external server.

Chapter 4 presents a summary of conclusions and contributions presented in this thesis.

CHAPTER 2

IMPROVED CONTINGENCY PLANNING VIA DISTINGUISHABILITY ANALYSIS OF OBSTACLE PREDICTIONS

2.1 Introduction

Motion planning is a fundamental component of nearly all robotic systems. It is relied upon to enable safe and satisfactory task completion by robots operating in a range of physical environments, from factories to homes [32].

Classical *deliberative* motion planning approaches, such as Dijkstra's method [16] and A* [27], can produce optimal paths, but typically consider only simple, unconstrained dynamic systems moving through static, deterministic environments. *Reactive* algorithms such as Vector Field Histogram [9] and the Dynamic Window Approach [18] do not produce globally optimal paths, but efficiently generate actions to locally avoid collision and seek out goals, while obeying dynamic constraints.

Most practical environments where robots are deployed are not static or deterministic, and this is especially true for road environments, which contain highly dynamic and unpredictable scenes due to the presence of traffic, pedestrians, and other road users. In these environments it is not possible to achieve satisfactory safety guarantees by assuming a static and deterministic environment, nor can such safety guarantees be provided by reactive motion planners. To successfully plan in road environments, the planner must anticipate the actions and intent of other vehicles and incorporate this information into the planning framework. As a simple example, merging into a lane autonomously is not

possible unless it is anticipated that other vehicles will yield to the merging car.

Many autonomous vehicles anticipate obstacle vehicles by predicting future motion, but importantly there exists little work that attempts to explicitly evaluate the correctness of these predictions as the scene evolves. This can lead to incidents such as the 2016 collision where Google's self-driving car mistakenly predicted that a transit bus would yield as the Google car moved into an adjacent lane [15]. Some systems rely on a sufficient re-planning frequency to account for changing obstacle predictions [40], but planning in this way is a reactive strategy that cannot be relied upon to provide safety guarantees, and performs increasingly poorly as vehicle speeds increase and obstacles are closer.

Most recent autonomous vehicles generate a set of trajectories from which a single path is driven [31]. This is analogous to how humans drive: we maintain multiple feasible paths at any given time to account for different possible scenarios. If road users only planned a single trajectory that could safely account for all possible obstacle behaviors, then vehicles would come to a standstill during any moderately complex interaction with other traffic. For this reason, human drivers and modern autonomous vehicles maintain *contingencies*—sets of trajectories that account for varying possible evolutions of the immediate road scene.

To enable successful autonomous vehicles, a flexible and robust planning framework is required, one that produces driving behaviors that are not unnecessarily conservative whilst providing guarantees on trajectory safety. To allow for safe, human-like driving behavior, the planning framework must consider an obstacle's predicted actions, much like human drivers do, maintain multiple paths, and determine the structure of, and inputs to, the planning process for each path.

In this chapter we outline a contingency-based motion planning framework for autonomous road vehicles. A multiple-model estimator is used to infer discrete obstacle actions from sensor observations, which in turn is used to inform contingency selection. We define a distinguishability metric that approximates the predicted probability of correctly identifying the discrete behavior of an obstacle from a set of possible actions. The metric is used to optimize expected cost and safety of contingency timing. The framework is designed to explicitly consider the fundamental relationship between perception and planning in autonomous vehicles.

Our approach is divided into four major subtasks:

1. Motion prediction of dynamic obstacles in the scene,
2. Determination of contingency plan timing,
3. Contingency plan generation,
4. Execution of the resulting plan, including selection of the optimal contingency.

This work focuses mostly on the second and fourth components, and, in particular, how contingency plan timing is dependent on the method by which the optimal contingency is selected at runtime. We first present related work on these subtasks, as well as existing planning frameworks that try to achieve similar goals.

2.2 Related Work

When planning in the presence of dynamic obstacles, the plan is influenced by the underlying predictions of the obstacles in the scene. There exists a large literature regarding motion prediction for dynamic obstacles in the context of autonomous driving.

Carnegie Mellon University's DARPA Urban Challenge 2007 entry, *Boss*, generated a deterministic prediction for each obstacle vehicle, by assuming vehicles will follow the lane while driving in structured environments, and by extrapolating vehicle speed and curvature in unstructured environments [17]. These predictions are used to evaluate candidate motion plans for safety. *Boss* used this planning strategy to great effect during the Urban Challenge, where the Carnegie Mellon team finished first. While incorporating obstacle predictions improved planning performance, the prediction model was limited to a single, deterministic hypothesis for the action of each obstacle vehicle, which does not account well for ambiguous, multi-modal obstacle predictions or even unimodal predictions with large uncertainty.

An intuitive and principled way to consider the possible motion of a dynamic obstacle is to encode the prediction in a probability distribution, as has been explored using continuous [26], discrete [20, 17], and hybrid [28] probability distributions.

Havlak and Campbell [28] employ a road-following controller and use a sigma point transform to propagate a Gaussian mixture predictive distribution, using a novel mixand splitting technique to control nonlinearities in the propagation step. This anticipation algorithm is only applied in some basic planning

scenarios, without any analysis of safety guarantees.

Hardy et al. present a nonparametric anticipation algorithm for road vehicles in [26], based on Gaussian process regression. In the work, the output from a data-driven model to predict driver inputs is fed through a physical vehicle model to generate predicted trajectories. The algorithm accurately predicts driver intent when navigating an intersection, but requires retraining for different types of intersections.

To plan over the resulting obstacle predictive distributions, a number of methods have been proposed in the literature. Bautin et al. [5], building on the work of Fraichard and Asama [19], use vehicle dynamic constraints and obstacle predictions to find regions in the state space where collision is inevitable with some probability, then plan over the state space, avoiding regions of inevitable collision. This technique produces a safe path with probabilistic guarantees on collision probability, and by using partial motion planning as in [39] can produce safe paths within a bounded time. However, the process of calculating the inevitable collision states is computationally expensive, and the resulting distributions describing these states are complex and difficult to plan optimally over.

Galceran, Cunningham, and others [20, 14] present a planning framework for autonomous vehicles, where every obstacle vehicle in the surrounding environment is characterized by a high-level, closed-loop policy, such as "follow lane", "change lane to the right", or "turn left". The likely policies for each vehicle are inferred from measurement history. Policy cost is assessed by sampling a policy for each vehicle in the scene and performing a closed-loop simulation, including basic modeling of vehicle interactions. This results in a decision making

framework that can anticipate obstacle vehicles reacting to its own policy. For example, vehicles are predicted to slow down to allow the ego vehicle to merge. Although the high-level policies are chosen so they are inherently safe, the sampling nature of the policy selection process cannot provide formal guarantees on collision probability.

Contingency planning, as introduced by Hardy and Campbell [25], refers to the generation of multiple plans that correspond to different hypotheses about the actions of dynamic obstacles surrounding the ego-vehicle. In most urban environments, planning a trajectory that assumes all obstacle hypotheses are simultaneously occurring leads to excessively conservative driving behavior. The authors are able to show that the contingency planning framework produces less conservative, more realistic driving behavior while providing guarantees on collision probability. However, the details of how contingencies are evaluated at runtime are not addressed, and the relationship between perception and planning is unexplored.

Hardy and Campbell define a fixed initial time interval where the contingency plans are identical, after which the plans diverge. Their formulation implicitly assumes that at the end of this initial time interval, sufficient information is available to estimate the true obstacle action with a high degree of certainty, and hence one of the available contingencies can be safely selected and driven. This assumption is clearly not valid generally, since many different vehicle maneuvers may initially appear similar, and the duration of apparent similarity is a function of the maneuvers themselves, and not a fixed interval. The result is that the planning approach is only valid for brief time windows and for certain initial conditions, outside of which the safety of the generated plans can not be

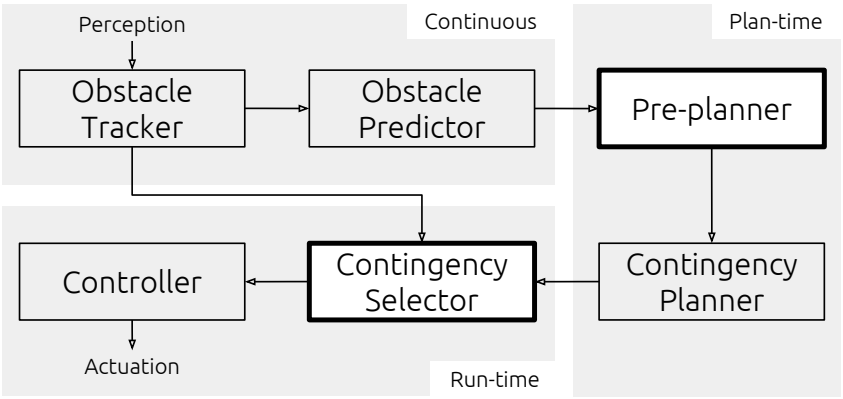


Figure 2.1: Architecture of the contingency planning system. The components shown in bold are the focus of the present work.

guaranteed.

2.3 Method

Notation used in this section is shown in Table 2.1.

2.3.1 System Architecture

The architecture of the planning system is shown in Figure 2.1. An *obstacle tracker* processes perception data to produce probabilistic state estimates of dynamic obstacles in the immediate environment. The *obstacle predictor* generates predictive state distributions for each obstacle at every timestep from the current time until the prediction horizon. There are a large number of existing methods for generating these predictions [26, 17]. The obstacle tracker is assumed to run continuously, in parallel to the planning and control components.

Table 2.1: Notation used in this section. Unless otherwise indicated, subscripts represent time or time ranges.

Vehicle	
Vehicle state at time index t	$\mathbf{x}_t$
Vehicle trajectory	$\boldsymbol{x}_{0:t}$
Vehicle measurement	$\mathbf{y}_t$
Vehicle measurement history	$\mathbf{y}_{0:t}$
jth discrete hypothesis, or *mode*	h^j
Number of discrete hypotheses	N_{hyp}
Prior probability of h^j	π^j
Posterior probability of h^j at t	μ_t^j
Planner	
Contingency start time / planning time	t_p
Contingency selection time	t_s
Contingency end time	T
Number of contingencies	N_{conting}
Shared trajectory	$\boldsymbol{x}_{t_p:t_s}$
Contingency plan for h^j	$\boldsymbol{x}^j_{t_s:T}$
Set of shared and contingency plans	$\boldsymbol{X}_{t_p:T}$
Optimization cost function	$J(\boldsymbol{X}_{t_p:T})$
Optimization constraint function	$\boldsymbol{C}(\boldsymbol{X}_{t_p:T})$

In this work we assume *hybrid* obstacle predictions, consisting of a discrete set of possible obstacle actions, each corresponding to a continuous predictive distribution.

The *contingency planner* produces an ego-vehicle trajectory for each relevant hypothesis about the evolution of the obstacles in the surrounding environment.

The *contingency selector* is responsible for the runtime decision of which plan to execute. The choice of plan is driven by current obstacle state estimates, used to produce an *a posteriori* distribution over obstacle actions. Further detail is given in section 2.3.4. The timing of this contingency selection process is determined by the *pre-planner*, which examines the *distinguishability* of the obstacle predictions—fundamentally, it identifies the expected time in the future where multiple hypotheses about the action of an obstacle are able to be disambiguated by the contingency selector. This component is described in Section 2.3.5, and the algorithmic details of the distinguishability analysis are presented in Section 2.3.6.

2.3.2 Obstacle Tracking and Prediction

In this work we assume that an object tracking algorithm exists which can perform measurement association, sensor fusion, and filtering to generate obstacle state estimates. Examples of such algorithms in the literature include those presented in [51] [34] and [12]. These object state estimates are used as inputs to the prediction step given be Equation 2.1.

The predictions used in this work are represented as a set of estimated state

distributions at each time index, from the current time until the end of the planning horizon. The predictive distribution for the state $\boldsymbol{x}_t^{\text{obst}}$ of a single obstacle at a single timestep t such that $t_p < t < T$ is given by:

$$p(\boldsymbol{x}_t^{\text{obst}} \mid \boldsymbol{y}_{0:t_p}^{\text{obst}}) = \int p(\boldsymbol{x}_t^{\text{obst}} \mid \boldsymbol{x}_{t-1}^{\text{obst}})\ p(\boldsymbol{x}_{t-1}^{\text{obst}} \mid \boldsymbol{y}_{0:t_p}^{\text{obst}})\, \mathrm{d}\boldsymbol{x}_{t-1}^{\text{obst}} \tag{2.1}$$

where $\boldsymbol{y}_{0:t_p}^{\text{obst}}$ is the obstacle measurement history until t_p. This is a very general formulation that makes no assumptions on the distribution or model type. Often, to make the prediction more tractable, assumptions about the distribution and model are made: for a linear model with Gaussian-distributed state and process noise this becomes a Kalman Filter prediction. The prediction for a non-linear transition function and arbitrary prior distribution can be approximated using Monte Carlo sampling methods.

Obstacle predictions are expressed as hybrid distributions—where the predictive distributions have both discrete and continuous components. The total predictive distribution at time t can then be decomposed as follows:

$$p(\boldsymbol{x}_t^{\text{obst}} \mid \boldsymbol{y}_{0:t_p}^{\text{obst}}) = \sum_{j=1}^{N_{\text{hyp}}} p(\boldsymbol{x}_t^{\text{obst}} \mid \boldsymbol{y}_{0:t_p}^{\text{obst}}, h^j)\ p(h^j \mid \boldsymbol{y}_{0:t_p}^{\text{obst}}) \tag{2.2}$$

where h^j represents a *discrete hypothesis* or *mode*. Each mode is predicted individually—the mode-conditioned prediction is analogous to Equation 2.1.

Each mode-conditioned predictive distribution is assumed to be a multivariate Gaussian over the state of the vehicle:

$$\boldsymbol{x}^{\text{obst}} \sim \mathcal{N}(\hat{\boldsymbol{x}}^{\text{obst}}, \Sigma^{\text{obst}}) \tag{2.3}$$

where the state is given by the following vector:

$$\boldsymbol{x}^{\text{obst}} = [x, y, \theta, v]^{\mathrm{T}} \tag{2.4}$$

the elements of which represent x position, y position, heading, and velocity, respectively.

Each obstacle mode is predicted using a simple road-following controller, based on the pure pursuit algorithm [13], which gives steering and throttle inputs to control a 2-D bicycle dynamic model. Different modes are predicted by following different road segments. The resulting controllers for each mode are nonlinear functions of the vehicle state. To propagate predictions through these nonlinear functions, we use a *sigma point transform* [49], which deterministically samples weighted points from the initial distribution, propagates the samples exactly using the nonlinear function, then fits a parametric distribution of the same type as the initial distribution to the transformed and weighted samples. Although this work uses single multivariate Gaussians to represent the predictive distribution, the sigma point transform is easily applicable to nonlinear propagation of a hybrid Gaussian mixture, as shown in [28]. Techniques also exist for filtering mixtures of Gaussians, such as the *blob* [42] and *Gaussian sum* [45] filters.

2.3.3 Contingency Planning

The contingency planner used in this work is an extension of the planner presented by Hardy and Campbell [25]. The planner aims to generate multiple trajectories for all possible evolutions of the scene, according to the discrete hypotheses of obstacle predictions, while providing for each trajectory formal guarantees on the probability of collision with an obstacle. The contingency planning problem is formulated as a simultaneous constrained optimization

over all contingencies, given by:

$$\begin{aligned} \min_{\boldsymbol{X}_{t_p:T}} \quad & J(\boldsymbol{X}_{t_p:T}) \\ \text{s.t.} \quad & \boldsymbol{C}(\boldsymbol{X}_{t_p:T}) \leq \boldsymbol{0} \end{aligned}$$

where $J(\cdot)$ is a cost function, $\boldsymbol{C}(\cdot)$ is a constraint function, and $\boldsymbol{X}_{t_p:T}$ is a set of all planned trajectories, equal to:

$$\boldsymbol{X}_{t_p:T} = \boldsymbol{x}_{t_p:t_s} \cup \{\boldsymbol{x}^j_{t_s:T}\}_{j=1}^{N_{\text{conting}}} \tag{2.5}$$

where the number of plans N_{conting} is equal to the number of discrete hypotheses N_{hyp} for the single-obstacle case. In general the number of contingencies is given by:

$$N_{\text{conting}} = \prod_{i=1}^{N_{\text{obst}}} {N_{\text{hyp}}}^i \tag{2.6}$$

where ${N_{\text{hyp}}}^i$ is the number of hypotheses associated with the ith obstacle, and N_{obst} is the number of obstacles.

The contingencies are structured such that until the *contingency selection time* t_s, the plans share an initial trajectory segment $\boldsymbol{x}_{t_p:t_s}$. After the selection time the plans diverge into N_{conting} trajectories, where $\boldsymbol{x}^j_{t_s:T}$ is the jth contingency trajectory.

The cost function $J(\cdot)$ contains terms penalizing trajectories with high acceleration, high curvature, close proximity to obstacles, and other comfort and safety metrics. The constraint function $\boldsymbol{C}(\cdot)$ ensures that each solution trajectory meets collision probability guarantees and obeys the dynamic constraints of the vehicle. The complete cost and constraint function expressions are given in [25]. For the purposes of assessing safety guarantees, we examine the collision probability of generated trajectories, and we use the total cost of path execution as a metric of (negative) path quality.

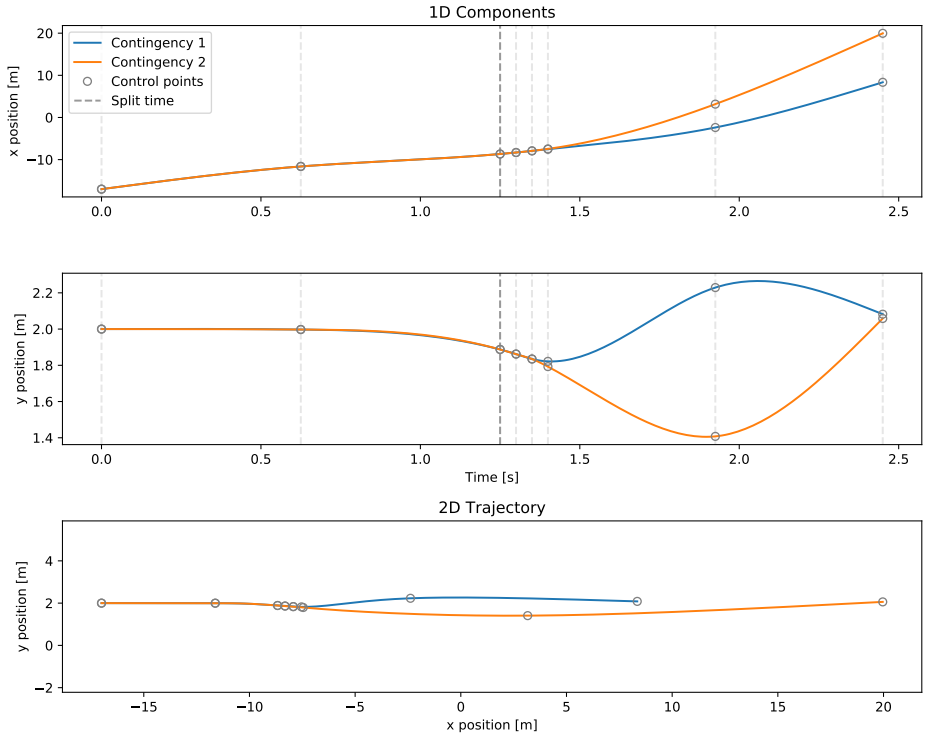


Figure 2.2: Spline-based trajectory representation. The control points for the cubic splines are shown as hollow circles. In this work the control points are not rigidly spaced in time, they are distributed according to the timing of the contingency selection. Additional control points are added after the selection time to assist in constraining the initial trajectory segments to be identical.

While planning the shared trajectory from the initial time t_p until the contingency selection time t_s, the true obstacle action is treated as unknown and all obstacle hypotheses are assumed to be simultaneously occurring. For planning purposes, it is assumed that after t_s the true hypothesis is known with certainty, and as such each plan only considers obstacle interaction costs and collision probability constraints that relate to its corresponding obstacle hypothesis.

A spline-based trajectory representation is used, as shown in Figure 2.2. Two cubic splines are used, for x-position and y-position, both expressed as a func-

tion of time. Before the optimization is performed, the control points of the cubic splines are specified at fixed points in time. The x and y values at these points are the only decision variables for the optimization, which reduces the dimensionality of the problem and decreases computation time. For the experiments in [25], the authors constrain the first two control points to be identical, so that the contingency plan has an initial fixed-length shared path segment before it splits into multiple trajectories, and the timing of the control points are equally spaced. However, in this work the timing of the contingency selection point is being studied, and as such the rigid timing of the control points in prior work is not suitable. We distribute the control points on either side of the contingency selection time, where the number of control points in the shared or separate contingency segments is proportional to the segment duration relative to the total trajectory duration. A number of additional shared control points are added immediately following the selection time, to further constrain the initial trajectory segments to be identical.

2.3.4 Contingency Selection

The process of determining which of the contingency plans should be chosen for execution is fundamental to the performance of any contingency planning framework. We term this process *contingency selection*. In this work, contingency selection is based on similar criteria to that of the trajectory generation process: the contingency chosen for execution is that which minimizes the *expected cost* of the path, whilst meeting *expected collision probability* constraints. Note that the trajectory constraints concerning vehicle dynamics are satisfied during planning and do not depend on obstacle vehicle observations, so only collision probabil-

ity constraints are considered during contingency selection. During runtime, contingency selection occurs after the start of plan execution, and is informed by obstacle measurements that occur between plan time t_p and selection time t_s.

The expected cost of each plan at selection time, based on measurements up to t_s, is given by:

$$\mathbb{E}\left[J(\boldsymbol{x}^j_{t_s:T})\right] = \sum_{k=1}^{N_{\text{hyp}}} p(h^k \mid \boldsymbol{y}^{\text{obst}}_{0:t_s})\; J(\boldsymbol{x}^j_{t_s:T} \mid h^k) \tag{2.7}$$

where $J(\boldsymbol{x}^j_{t_s:T} \mid h^k)$ is the cost of the trajectory considering only hypothesis h^k. The expected collision probability of a plan is defined similarly:

$$\mathbb{E}\left[\mathbb{P}_{\text{coll}}(\boldsymbol{x}^j_{t_s:T})\right] = \sum_{k=1}^{N_{\text{hyp}}} p(h^k \mid \boldsymbol{y}^{\text{obst}}_{0:t_s})\; \mathbb{P}_{\text{coll}}(\boldsymbol{x}^j_{t_s:T} \mid h^k) \tag{2.8}$$

where $\mathbb{P}_{\text{coll}}(\boldsymbol{x}^j_{t_s:T} \mid h^k)$ is the collision probability of the trajectory, considering only obstacle motion as predicted by hypothesis h^k. A tight upper bound on collision probability between the ego and obstacle vehicles is calculated using the algorithm presented in [25].

The posterior mode probability term $p(h^k \mid \boldsymbol{y}^{\text{obst}}_{0:t_s})$ in Equations 2.7 and 2.8 can come from any Bayesian multiple-model tracking algorithm. Here we use a standard non-interacting multiple-model estimator formulation [4], where each mode-conditioned estimate is maintained recursively by an independent filter. This filter formulation assumes that the observed system does not exhibit mode-switching, which reflects the assumptions made during obstacle prediction of a single, fixed mode per hypothesis. The posterior mode probabilities are notated as:

$$\mu^j_t = p(h^j \mid \boldsymbol{y}^{\text{obst}}_{0:t}) \tag{2.9}$$

and are updated recursively according to:

$$\mu^j_t \propto p(\boldsymbol{y}^{\text{obst}}_t \mid \boldsymbol{y}_{0:t-1}, h^j)\; \mu^j_{t-1} \tag{2.10}$$

where $p(\boldsymbol{y}_t^{\text{obst}} \mid \boldsymbol{y}_{0:t-1}, h^j)$ is the mode-conditioned measurement innovation. This update is exact for linear-Gaussian assumption, where the true underlying mode does not switch and a Kalman filter is used for each mode. In this work, however, a sigma point filter [49] is used to maintain approximate mode-conditioned estimates for each hypothesis, so the overall multiple-model estimates are not exact.

2.3.5 Pre-planning

It is apparent that the expected cost and collision probability of a selected contingency is highly dependent on the posterior mode probabilities over obstacle actions at selection time. Specifically, if the distribution of mode probabilities does not strongly support the true obstacle action, then the selected contingency is likely to exceed collision probability constraints, as obstacle hypotheses that conflict with the chosen path may have a non-negligible probability of occurring. To guarantee trajectory safety of the complete planning framework, the timing of contingency selection must be considered *before* the trajectories are planned, based on the predicted actions of the obstacle vehicle.

The pre-planner chooses the contingency selection time to be *the earliest time where the relevant obstacle hypotheses are distinguishable to a specified confidence level*. This strategy encodes the intuitive notion that earlier decision making and reaction is preferable in a road environment, while ensuring that the ego vehicle does not react before it is sufficiently confident in its predictions of nearby vehicles. The distinguishability confidence level is the probability that an obstacle hypothesis will be incorrectly identified at the contingency selection time,

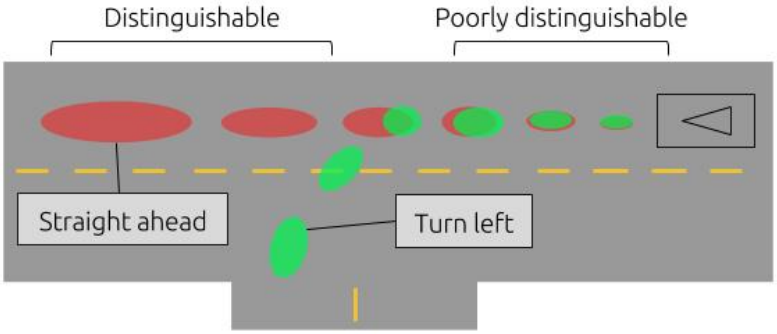
which is a design parameter can be tuned to provide desired probabilistic safety guarantees. Because the contingency selection time must be determined prior to trajectory optimization, the future distinguishability of obstacle modes is estimated via analysis of the corresponding predictive distributions.

2.3.6 Prediction Distinguishability

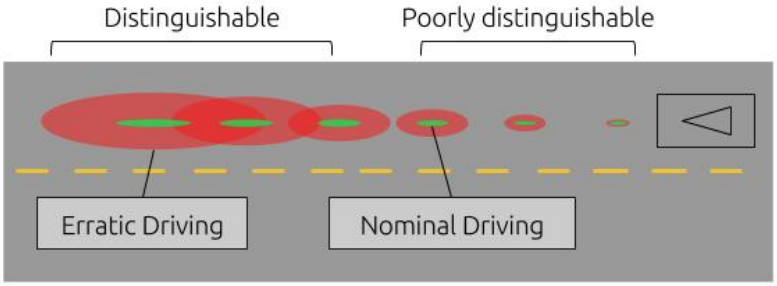
The *distinguishability* of predictions can be informally defined as the degree to which the true discrete hypothesis can be identified from future measurements. Intuitively, predictions are poorly distinguishable when different discrete hypotheses have largely overlapping predictive distributions, as shown in Figure 2.3. In the figure, the distributions are clearly more separable toward the end of the prediction horizon, and the discrete hypotheses can be distinguished.

Discrete hypotheses could correspond to high-level driver actions—such as turning left at an intersection or changing lanes—or, more generally, any sufficiently distinguishable decomposition of a continuous predictive distribution. This partitioning could be based on high-level obstacle intent (e.g. obstacle turns left or continues straight), by analysis of obstacle behavior (e.g. erratic or predictable), or by considering impacts on the ego vehicle trajectory (e.g. obstacle stops at stop line, or encroaches into ego lane). Examples are shown in Figure 2.3.

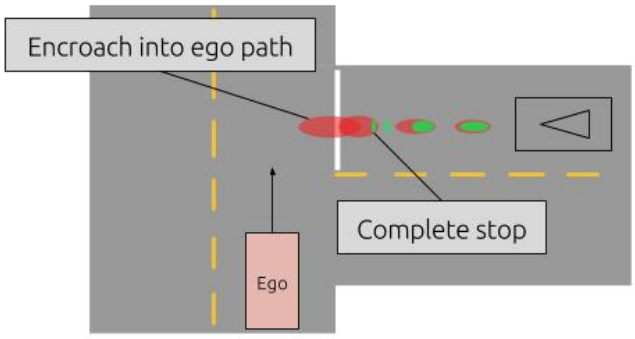
We define the distinguishability of discrete hypotheses as the expected probability that, at a given time, the maximum a-posteriori (MAP) hypothesis estimate is the true obstacle action. For the purposes of distinguishability analysis, the contingency selection task is simplified to a hard classification of the poste-



(a) Discrete hypotheses correspond to high-level obstacle intent.



(b) Discrete hypotheses correspond to obstacle driving behavior.



(c) Discrete hypotheses correspond to level of interaction with ego trajectory.

Figure 2.3: Qualitative examples of partitioning a continuous predictive distribution based on distinguishability. The red and green ellipses represent the predictive distributions for each discrete hypothesis, equally spaced in time over the prediction horizon.

rior mode distribution to a single hypothesis. This simplification is reasonable, as the purpose of the distinguishability analysis is to determine the contingency selection time—a fixed point in time by which a selection must be made with certainty. Closed-form calculation of distinguishability as defined above is not possible; a Monte Carlo estimate is required, where the prediction model and multiple-model estimator are simulated in tandem.

To reduce the need for expensive and approximate Monte Carlo simulation of the multiple-model estimator, a conservative approximation to the true distinguishability is used, where the MAP classifications from the full multiple-model estimation process are replaced with single-timestep classification based on the predictive distribution. The classification is "single-timestep" in the sense that for the calculation of distinguishability at a given time, only the predictive distributions at that same time instant are used. However, during preplanning, this distinguishability calculation is repeated for each timestep within the prediction horizon to determine the contingency selection time.

Optimal classification of a measurement $\boldsymbol{y}_t$ for $t > t_p$ is given by the Bayes decision rule:

$$\bar{h} = \underset{h^k}{\operatorname{argmax}} \ \pi_k \ p(\boldsymbol{y}_t \mid \boldsymbol{y}_{0:t_p}, h^k) \tag{2.11}$$

where π_k is the prior probability of hypothesis h^k, and the mode-conditioned *predicted measurement distribution* $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{0:t_p}, h^k)$ can be obtained from a measurement model $p(\boldsymbol{y}_t \mid \boldsymbol{x}_t)$ and predictive distribution $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{0:t_p}, h^k)$ as follows:

$$p(\boldsymbol{y}_t \mid \boldsymbol{y}_{0:t_p}, h^k) = \int p(\boldsymbol{y}_t \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t \mid \boldsymbol{y}_{0:t_p}, h^k) \ \mathrm{d}\boldsymbol{x} \tag{2.12}$$

When using the Bayes decision rule, the misclassification probability, or *error probability*, of measurements sampled from the entire predicted measurement

distribution is given by:

$$\mathbb{P}_t^e = 1 - \int \max_k \left\{ \pi_k p(\boldsymbol{y}_t \mid \boldsymbol{y}_{0:t_p}, h^k) \right\} \mathrm{d}x \tag{2.13}$$

The expression in Equation 2.13 reduces to a one-dimensional Gaussian CDF for the two-class problem where the class-conditional densities are Gaussians with identical covariances. In the general case, however, there does not exist a closed-form solution, and Monte Carlo sampling is slow and does not provide a definite bound. Bhattacharya and Toussaint [7] present a general upper bound on the error probability in terms of the *affinity* ρ:

$$\mathbb{P}_t^e \leq \frac{N_{\text{hyp}} - 2}{N_{\text{hyp}} - 1} + \frac{1}{N_{\text{hyp}} - 1} (\pi_1 \pi_2 \cdots \pi_{N_{\text{hyp}}})^{1/N_{\text{hyp}}} \rho(h^1, \ldots, h^{N_{\text{hyp}}}) \tag{2.14}$$

where the general expression for the affinity of a set of hypotheses is:

$$\rho(h^1, \ldots, h^{N_{\text{hyp}}}) = \int \left[p(\boldsymbol{x}_t \mid \boldsymbol{y}_{0:t_p}, h^1) p(\boldsymbol{x}_t \mid \boldsymbol{y}_{0:t_p}, h^2) \cdots p(\boldsymbol{x}_t \mid \boldsymbol{y}_{0:t_p}, h^{N_{\text{hyp}}}) \right]^{1/N_{\text{hyp}}} \mathrm{d}\boldsymbol{x}_t \tag{2.15}$$

The affinity for the two-hypothesis case, also known as the *Bhattacharyya coefficient*, is straightforward to compute for multivariate Gaussians:

$$\rho(h^j, h^k) = \frac{1}{8} (\hat{\boldsymbol{y}}_j - \hat{\boldsymbol{y}}_k)^T \Sigma^{-1} (\hat{\boldsymbol{y}}_j - \hat{\boldsymbol{y}}_k) + \frac{1}{2} \ln \left(\frac{\det \Sigma}{\sqrt{\det \Sigma_j \det \Sigma_k}} \right) \tag{2.16}$$

where $\hat{\boldsymbol{y}}_i$ and Σ_i are the means and covariances of the distributions, and

$$\Sigma = \frac{\Sigma_j + \Sigma_k}{2}. \tag{2.17}$$

Using Equations 2.14 and 2.16, an efficient upper-bound on the simple classification error probability between two discrete hypotheses can be found. The bound has been extended to the multi-class case in [43].

By treating the estimation process as a single-timestep classification for the purposes of distinguishability analysis, the measurement history is effectively

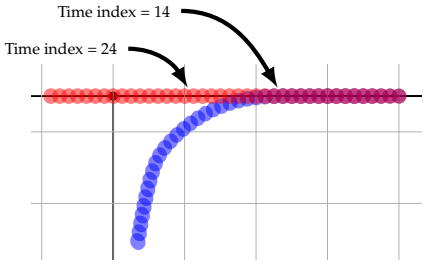
ignored. By contrast, the multiple-model estimator that is actually employed to estimate hypotheses at runtime recursively operates on the complete observation history. For this reason, using the error probability of the single-timestep classification leads to a a conservative estimate of the true distinguishability, but provides a fast and closed-form bound.

2.4 Results

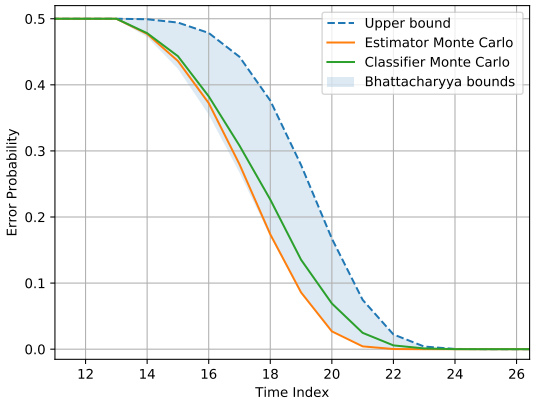
Simulations were developed in the Python language, with cost and constraint calculations written in C++ to reduce computation time. The NLOPT library [30] was used for contingency optimization, and the FADBAD++ automatic differentiation library used for calculation of derivatives [6]. Simulations were executed on a notebook Linux computer. The trajectory optimization took approximately 30 seconds to complete for two contingencies, although it was not heavily optimized for performance. The distinguishability metric presented here is not computationally expensive—calculation between two hypotheses over 75 prediction timesteps takes approximately 30 ms—and can be readily optimized further by implementing in fast linear algebra libraries, such as Eigen [23].

2.4.1 Prediction Distinguishability

To evaluate the proposed distinguishability metric, we performed Monte Carlo trials of the state propagation and estimation process. The evolution of an obstacle vehicle was predicted, according to an discrete action sampled from the prior distribution. Process and measurement noise was added while generating



(a) Positional components of predicted measurement distributions (2-σ ellipses). Heading and velocity states not shown. The vehicle starts at right, and moves left and down, respectively for the two hypotheses.



(b) Error probability when estimating the mode of two obstacle trajectories over time. Values only shown for a subset of the full prediction time range. Error probabilities for a multiple-model estimator and a simple classifier are shown, in addition to the Bhattacharrya upper and lower bounds on simple classification error probability.

Figure 2.4: Monte Carlo trials of the distinguishability of simulated turning and non-turning vehicles.

the simulated state and measurements. The multiple-model estimator was run online using simulated measurements. Posterior mode probabilities from the estimator were used to assess the most likely obstacle hypothesis. The estimator error probability (the true distinguishability measure) was compared to both the proposed Bhattacharrya upper bound (Equation 2.16), and the Monte Carlo sampled single-timestep classification error probability.

Figure 2.4.1 shows the results of 15 000 Monte Carlo simulations of the scene evolution and estimator operation, each over 40 timesteps of 25 ms. The x-y position components of the predictive distributions are shown in the first figure, where the discrete hypotheses are either straight driving or a left turn. The second figure shows the resulting error probabilities over the simulation period.

The estimator error probability is generally below that of the simple classifier. This result is intuitive, as the simple classifier does not consider the measurement history and state evolution, whereas the estimator uses an iterative algorithm that incorporates time information. The fast distinguishability metric used to inform planning in this work, the Bhattacharyya upper bound on simple misclassification probability (Equation 2.16), is a much more conservative metric than the true estimator error probability. Despite the conservatism of this value, our subsequent planning results show that plans structured using this distinguishability metric meet safety and dynamic constraints during execution, while not producing overly cautious driving behavior.

2.4.2 Driving Scenario

Experiments were conducted using the proposed planning framework in a simulated driving scenario where two vehicles are approaching a 3-way intersection from opposing directions. The scenario is shown in Figure 2.5. The ego vehicle is approaching from the left, intending to travel straight through the intersection. The obstacle vehicle is approaching from the right, and can either travel straight through or turn right at the intersection. It is assumed that the obstacle vehicle has the right of way regardless of which action it takes, and it takes each action with equal prior probability. The obstacle vehicle state is simulated using a road-following controller, with white Gaussian process noise injected to simulate uncertain dynamics of the system. White Gaussian measurement noise is also added to the simulated full-state measurements of the obstacle vehicle, which are consumed by the ego vehicle planning framework for obstacle tracking and contingency selection.

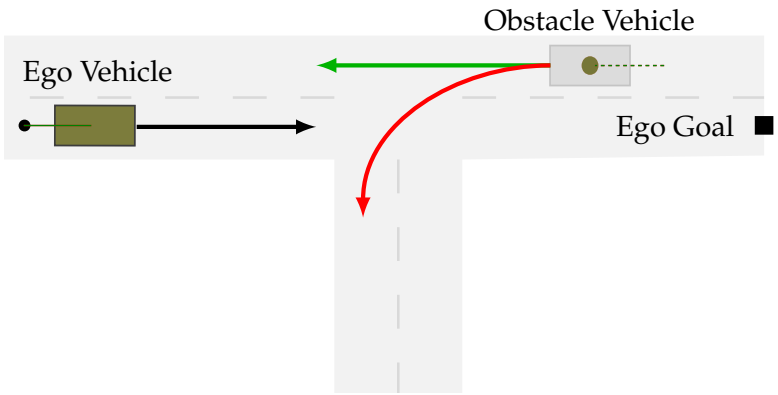
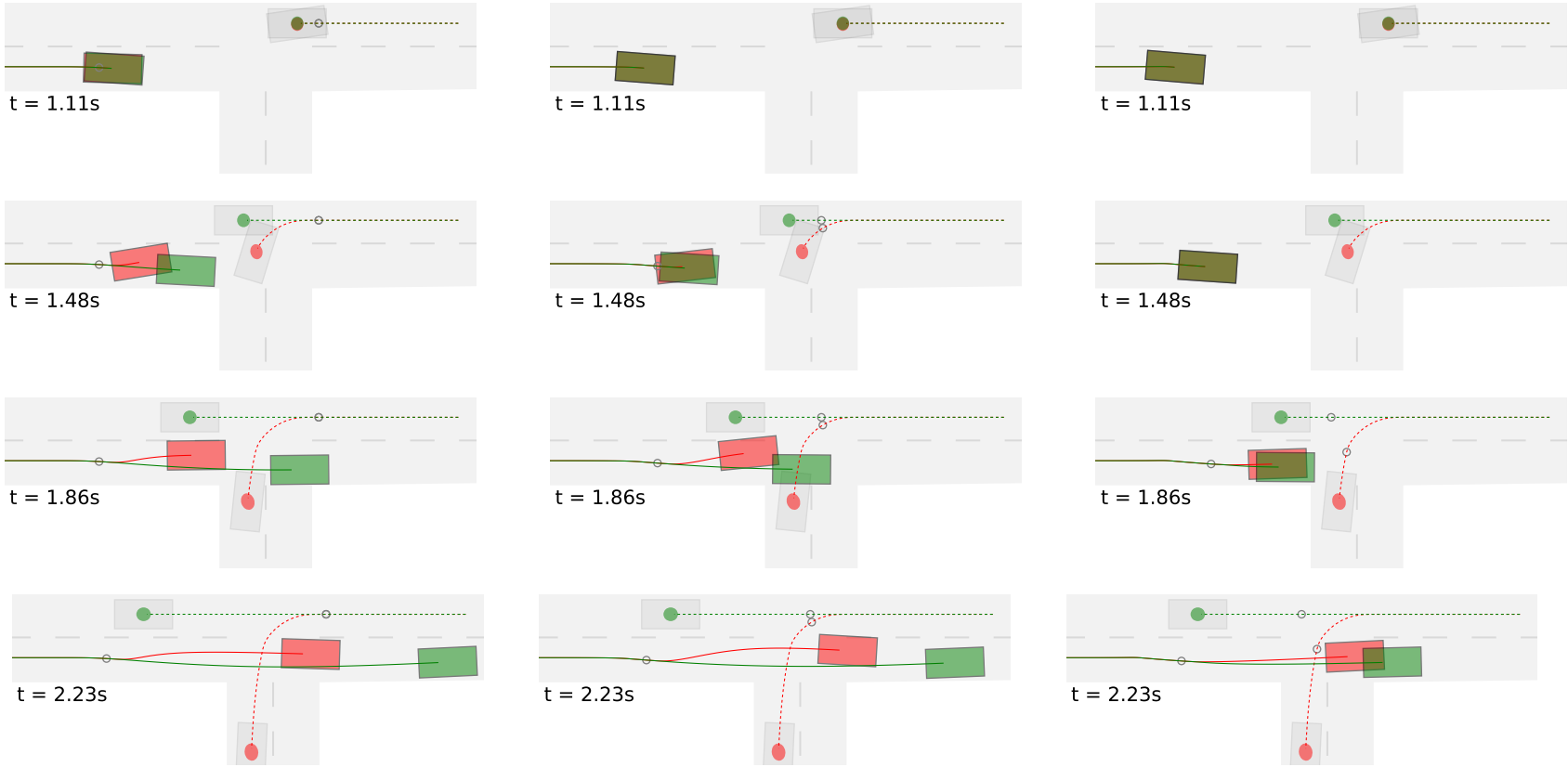


Figure 2.5: Driving scenario for planning experiments. The obstacle vehicle has two possible discrete actions, traveling straight through the intersection or turning left. The ego vehicle must yield to the obstacle vehicle at all times, while attempting to reach its planning goal by moving straight through at the intersection.

Planned Trajectories

Frames from planned trajectories are shown in Figure 2.6. Three different contingency selection times are used for planning: at, before, or after the approximate optimal selection time in a minimum-expected-cost sense. The figure shows the obstacle predictions and the corresponding generated trajectories.

When the plan is generated with a contingency selection time that is earlier than optimal, the individual trajectories are able to diverge sooner and "react" to each obstacle hypothesis individually. These individual trajectories can be less aggressive in terms of lateral acceleration and curvature, while still having low distance-to-goal penalties, producing paths with low overall cost *at planning time*. Importantly, at selection time, the *expected execution cost* of the path



(a) Early contingency selection (b) Optimal contingency selection (c) Late contingency selection

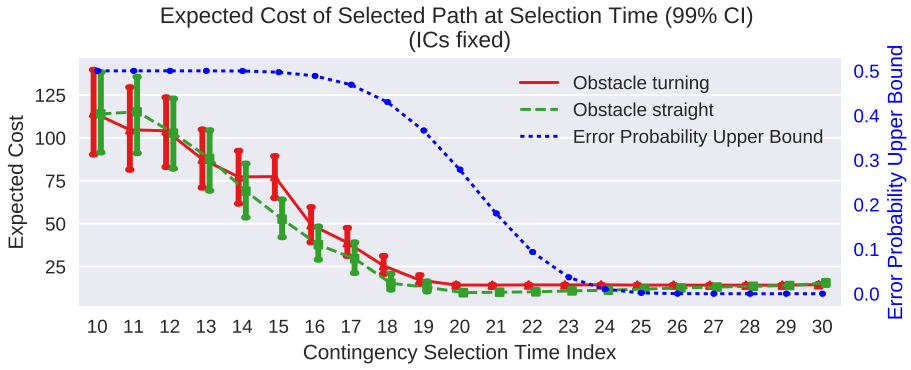
Figure 2.6: Example contingency plans generated for different contingency selection timings. The ego vehicle contingency plans are shown in green and red, starting at left. The similarly-colored ellipses indicate the covariance of the obstacle prediction corresponding to the matching ego trajectory, at the current time. The vehicle location at contingency selection time is indicated by hollow circles on both the ego and obstacle paths.

is much higher as there is still significant uncertainty about the action of the obstacle. The execution cost is dominated by penalties associated with collision probability and proximity to obstacles. These results are reflected in Monte Carlo simulations detailed in Section 2.4.2.

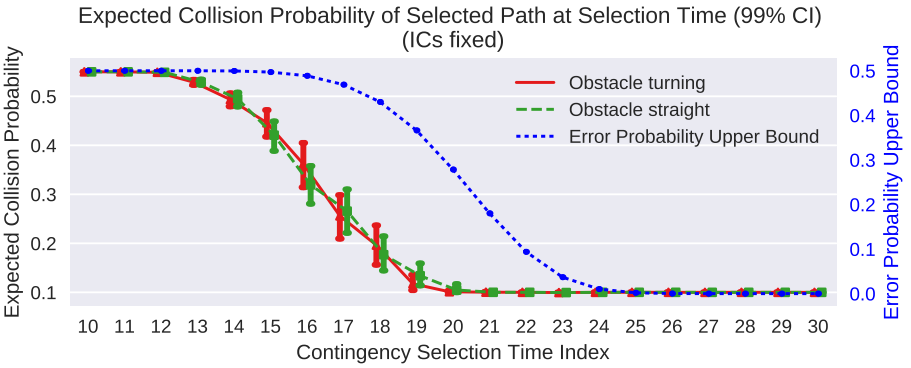
The later the contingency selection time, the longer the initial segment of the trajectory that is shared between all contingencies, and hence a greater duration of the trajectory must be planned under the conservative assumption that all obstacle hypotheses are simultaneously occurring. It follows that when the selection time is later, there is a shorter period where the planner can leverage knowledge of the true obstacle mode to generate a less conservative path. In Figure 2.6, this difference is most visible in the relative final positions of the ego vehicle between the two contingencies. The difference in position is larger for an earlier selection time, as the trajectory can diverge earlier to more aggressively avoid obstacles.

Monte Carlo Experiments

For each of the experiments the obstacle vehicle is simulated until the contingency selection time, at which point the expected cost and collision probability of each contingency plan is evaluated and a path chosen according to Section 2.3.4. In our framework, prediction and planning are deterministic, so these steps are performed once at the beginning of a large number of Monte Carlo simulations of the obstacle state evolution and estimator dynamics. An unrealistically high maximum collision probability constraint of 0.1 was chosen to generate interesting vehicle interactions, and is also necessary because the noise and prediction models used in the obstacle predictor and contingency selector



(a) Expected cost



(b) Expected collision probability

Figure 2.7: Expected cost and expected collision probability of chosen contingency at selection time, for trajectories planned with varying contingency selection time. The distinguishability of obstacle hypotheses at the given time index is indicated by the error probability.

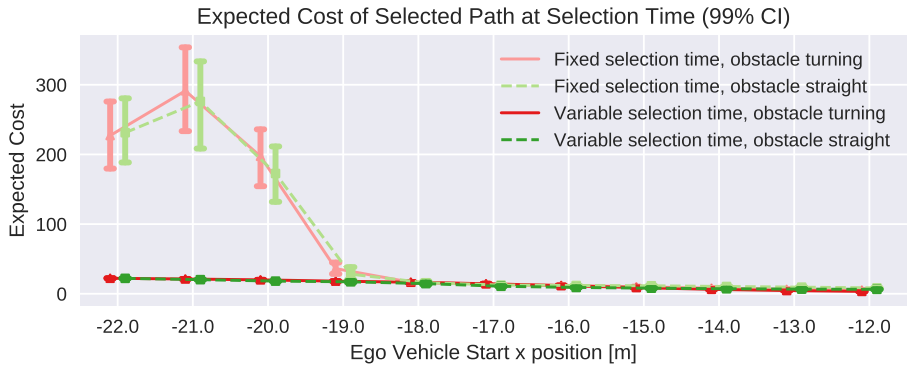
are not well calibrated to real-world driving examples.

Figure 2.7 shows the results of 20 000 Monte Carlo trials each for manually specified selection times ranging from timesteps 20 to 30 (a 0.25 s range). In the expected cost results, it can be clearly seen that selecting a contingency too early results in a selected path with very high cost, as ambiguity among obstacle hypotheses is not sufficiently resolved at the selection time, i.e. the selected contingency was planned only considering interactions with a single discrete

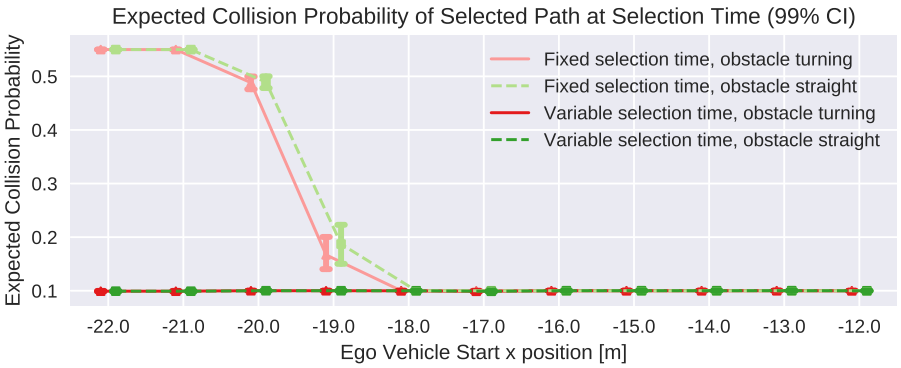
hypothesis, but because there remains some significant probability ascribed to other hypotheses at the selection time, the expected cost of taking any path is large. A different effect dominates for later contingency selection times, where the true obstacle hypothesis is estimated with high probability but the trajectories that wait longer to diverge are more conservative than necessary. This effect is more marked for the trajectory corresponding to the obstacle driving straight through the intersection, as the planner can sooner discount the possibility of the obstacle turning across the ego trajectory, and the trajectory accelerates earlier to move closer to the goal. By contrast, the second trajectory will have to yield to the turning vehicle regardless of when the contingencies diverge and, as such, the cost is relatively constant even for very delayed selection timing.

The expected collision probability results in Figure 2.7(b) show a similar result. At planning time each contingency meets collision probability constraints, because perfect knowledge of the true obstacle mode is assumed. However, when a contingency is selected prematurely at runtime, the contingency selector still has some uncertainty regarding the action of the obstacle, and obstacle modes that would conflict with the selected ego trajectory still have small, but significant, posterior probability. As a result the expected collision probability of the path, as calculated at selection time, often exceeds safety constraints.

Also shown in Figure 2.7 is the proposed fast distinguishability metric, calculated at each time index between the two obstacle hypotheses. When choosing the selection time during the planning phase, we approximate the optimal selection time as the earliest time index where the predictions are sufficiently distinguishable to a level determined by a threshold parameter. According to the criteria for contingency selection—the path with the lowest expected cost



(a) Expected cost



(b) Expected collision probability

Figure 2.8: Expected cost and expected collision probability of chosen contingency at selection time, for trajectories planned with fixed and distinguishability-based selection times. Values shown for Monte Carlo trials under varied initial conditions of ego and obstacle vehicles.

that also meets expected collision probability constraints—in this scenario the true optimal selection time index is 25. Even using a very conservative error probability threshold of 0.0005 , the proposed distinguishability-based method gives a selection time index of 27, close to the optimal selection time.

A second experiment to assess the robustness of the proposed framework involved varying the initial conditions of both the obstacle and ego vehicle. Both vehicles' starting locations were varied along their path of travel, but in

opposing directions such that they would both arrive at the intersection at approximately the same time and produce interesting interactions. The goal of this experiment was to assess the robustness of a planner using a distinguishability-based selection time against a naïve contingency planner using a fixed selection time, such as that presented by Hardy and Campbell [25]. Figure 2.8 shows the expected cost and expected collision probability of the selected contingency across different initial conditions, for each planner and each true obstacle action. The results clearly show that while using a fixed selection time will occasionally match or slightly improve on plan cost as compared to a variable selection time, the expected cost and collision probability of paths are not consistent when the initial conditions change, and are generally much higher than for distinguishability-based selection timing. Importantly, safety guarantees are often not met when executing trajectories generated by the fixed selection time planner.

2.5 Conclusion

This chapter presents a general motion planning framework for autonomous vehicles operating in environments with obstacles that have uncertain intent. The method considers the dynamics of the process by which discrete obstacle actions are estimated, generating trajectories with a high probability of safe execution. These trajectories are lower-cost and safety-guaranteed as compared to paths from a naïve contingency planner that ignores prediction structure and the characteristics of the vehicle perception system.

A fast upper bound on prediction distinguishability is presented, which can

be used to efficiently find future points in time where discrete predicted obstacle actions are able to be disambiguated by sensor measurements. Contingency selection occurs at the earliest time where obstacle hypotheses are sufficiently distinguishable. Even though this distinguishability metric is a conservative upper bound, planning simulations show that even using a very low threshold on prediction distinguishability generates plans that improve safety and path quality, while avoiding overly cautious driving behavior. The metric can be applied to compare more expressive distributions, such as multiple Gaussian mixtures, with minor modification.

In experimental results, only hybrid obstacle predictions with two discrete hypotheses are studied. The multiple-model estimator is readily scalable to a larger number of hypotheses, with computation increasing linearly as the number of models increases. The distinguishability metric is also efficiently applicable to the multiple-hypothesis case, since the same metric can be used to calculate *one-vs.-one* error probabilities between each pair of hypotheses, resulting in $N_{\text{hyp}}(N_{\text{hyp}} - 1)/2$ evaluations of the metric at each prediction timestep. Although this increases quadratically with the number of hypotheses, the calculation is independent for each obstacle, and only needs to be evaluated for each timestep until the distinguishability threshold is reached. Furthermore, any indistinguishable hypotheses can be clustered together, as in [24], reducing the number of contingencies that need to be generated—the dominant computational expense in the contingency planning framework—and mitigating the otherwise exponential increase in contingencies with the number of obstacles and hypotheses.

The concept of hypothesis distinguishability is not limited to the case where

hypotheses correspond to driving maneuvers, such as changing lanes or turning left at an intersection. As discussed above, discrete hypotheses can be any decomposition of the predictive distribution such that the components are predicted to be distinguishable at a future time. Further investigation is required into how these hypotheses might be automatically generated from a given complex distribution. Automatic generation of prediction hypotheses would also increase the usefulness of black-box prediction models, where rich continuous distributions are provided without explicit discrete partitioning, such as in deep learning or Gaussian process based models [26].

CHAPTER 3

SYSTEM ARCHITECTURE FOR SECURE AUTONOMOUS VEHICLES

3.1 Introduction

Our modern world is increasingly populated by networked devices containing high-performance computing, complex sensor interfacing, and significant physical actuation components [41]. As systems of this nature become more advanced and more connected, using current software and hardware design principles almost inevitably exposes serious security flaws, as recent experience indicates [46]. Systems such as automobiles and aircraft have significant potential for harm if they are maliciously compromised, and as such it is increasingly important to secure these systems against attack.

Recent studies have revealed significant attack surfaces across infotainment, wireless internet, and diagnostics components in modern automobiles, allowing an attacker to remotely control steering, brake, and throttle [11, 33]. With the widespread development and increased deployment of driver assistance and self-driving features [8], in the near future we can expect greater vehicle connectivity, more sensing modalities, a larger volume of data from external sources, and a higher degree of automated actuator control. These factors increase both the vulnerability of the system to malicious interference, and also the consequences of such an attack.

In the computer security literature there exist software languages that enforce separation of information with differing confidentiality and integrity levels [44]. These languages are provably robust to certain classes of attacks com-

monly used to take control of cyber-physical systems. Similarly, modern hardware design techniques can produce processor architectures that prevent or limit any interference between one (possibly untrusted) process and another running on the same processor [50]. Combining modern information security techniques such as these at the system design level is critical for ensuring security for complex devices.

In the autonomous car example, it is difficult or impossible to completely isolate networked or non-critical components from safety-critical systems. Autonomous perception, navigation, and control algorithms require data from myriad sources, such as online maps [1], wireless communication with nearby vehicles [2], and onboard sensors. Malicious modification of any of these external or unreliable input data sources can result in large changes to driving behavior. Only through system-level design principles, and hardware, software, and algorithm verification can these systems be made provably secure.

This chapter introduces a prototype system architecture for a verifiably secure autonomous vehicle. It is implemented on a mobile robotic platform and experiments are performed to simulate a remote attack scenario. A map verification algorithm is used to verify external data coming from an untrusted source. The planning and map verification software is developed with existing tools that enforce *information flow control* at the language level. Extensions of the platform to support more fully-featured autonomous driving are discussed, as are analogies between the prototype architecture and hardware and software components on real-world autonomous vehicles.

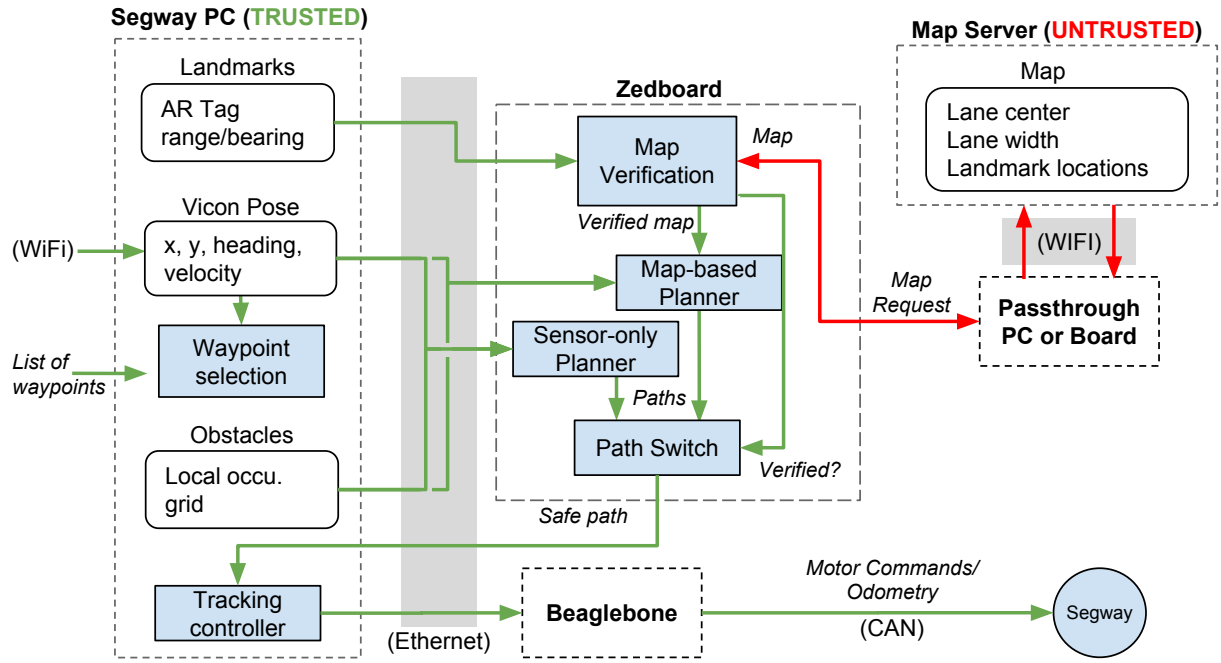


Figure 3.1: Architecture of the demonstration system. Green and red arrows represent trusted and untrusted information flows, respectively. Dashed boxes indicate separate physical computing units.

3.2 System Architecture

The system architecture is intended to represent the primary algorithmic and software components of an autonomous vehicle: sensor processing, path planning, and wireless communication. It is shown in Figure 3.1.

The architecture is implemented on a Segway mobile robotic base, using sensor input from LIDAR and camera, and pose estimates from a Vicon motion tracking system. The path planner and map verification are implemented on a Zedboard system-on-a-chip development board. An external map server communicates with the Zedboard via a wireless internet connection. Most sensor

processing and motor control functions are implemented on a mobile computer onboard the robot (the *Segway PC*).

3.2.1 Threat Model

The potential attacks that this architecture is designed to mitigate are of the form where an untrusted hardware or software component is able to influence a critical driving function of the robot. This could arise where a software component is internet-connected, or otherwise vulnerable to injection of malicious code, and is also connected to a critical driving software component. We assume that critical software components are not able to be directly compromised, although their inputs may be manipulated if an untrusted source is attacked. In a modern automobile, such an attack could involve compromising the infotainment system and thus gaining access to steering, brake, and throttle controls via the vehicle CAN network.

The map server used in these experiments represents an external untrusted component, but the security considerations are equivalent for the case where a local software component is untrusted. If the untrusted software is running on the same processor as safety-critical computations, a further class of hardware attacks are possible, including timing channel attacks [22] and other attacks exploiting shared hardware resources across tasks [29].

3.2.2 Sensor Suite

A multi-modal sensor suite is used to approximate the range of sensors commonly found on autonomous vehicle platforms. A LIDAR sensor is used for local obstacle avoidance, its measurements incorporated into the path planning algorithm. Measurements from this sensor are fused online to create an occupancy grid of the surrounding environment, indicating the probability that a grid cell contains an obstacle [48]. The occupancy probability of the grid cells, in addition to the lane information from a map, constitute the planner cost function (presented in more detail in Section 3.2.4).

An external Vicon motion tracking system is used to provide sub-centimeter accurate robot pose estimates to the system, the *Vicon Pose*. This is analogous to a Global Navigation Satellite System (GNSS)—for example the Global Positioning System (GPS)—receiver that is present in most modern vehicles. Both systems provide absolute pose of the robot, although an indoor motion tracking system typically has much higher accuracy and measurement frequency than a GNSS receiver. For these reasons it is used to also provide feedback to the low-level path-following controller, although in a typical autonomous vehicle a more sophisticated algorithm, such as a particle filter [35], would be used to augment GNSS measurements and localize the vehicle in its immediate environment with high accuracy.

A camera is used to detect visual landmarks in the field of view of the vehicle. The landmarks are intended to correspond to prominent visual features in road environments, such as road signs, stop lines, and other easily and repeatably detectable objects in the scene. Such landmarks are expected to be provided as part of the external map information, to enable verification of the map

using sensor-reported landmark measurements. To simplify the image processing tasks in our experiments we use *Aruco tags* [21], robust fiducial markers for which there is existing open-source detection software. For example, the Aruco (AR) tag detection library in OpenCV[1] reports, for each tag in the camera field of view, a 6 degree-of-freedom transform between the camera frame and the tag.

3.2.3 Map Verification

An external map server provides the vehicle with a two-dimensional map of landmarks, lane positions, and other planning information. The landmark locations are each a single point describing some sensor-visible feature in the environment, and the lane information is a sequence of line segments accompanied by planning cost information (see Section 3.2.4 for details).

Map information is provided dynamically in segments covering the immediate area around the robot. The robot queries the map server for new map segments when it has moved close to the edge of the current map segments. While it would be straightforward to save the map in memory on the robot, given the size of our experimental scenarios, for autonomous vehicles operating in large-scale environments this is generally not possible and maps must be downloaded, typically continuously, from an external source.

We treat the external map server or, equivalently, the communication channel to the map server, as untrusted. In order to use the map for planning, we require independent verification of the map information against local sensor data. To achieve this, we assume a Gaussian measurement distribution around each

[1]OpenCV, `www.opencv.org`

expected landmark location, as reported by the map, and assess the agreement with local sensor measurements using a statistical hypothesis test.

First the test statistic is calculated for a measurement $\boldsymbol{y} \in \mathbb{R}^2$ consisting of x and y landmark position in map coordinates:

$$Z = (\boldsymbol{y} - \boldsymbol{m}_0)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{m}_0) \tag{3.1}$$

which follows a χ^2 distribution under the null hypothesis that the measurement $\boldsymbol{y}$ is sampled from the expected Gaussian measurement distribution:

$$p(\hat{\boldsymbol{y}}) = \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{\Sigma}) \tag{3.2}$$

where $\hat{\boldsymbol{y}}$ is the expected measurement, $\boldsymbol{m}_0$ is the map-reported landmark location, and the covariance of the distribution is assumed to be circular and calculated as follows:

$$\boldsymbol{\Sigma} = \left(\sigma + \alpha \left\| \boldsymbol{m}_0 - \boldsymbol{x}^{\mathrm{robot}} \right\|\right) \boldsymbol{I} \tag{3.3}$$

where $\boldsymbol{x}^{\mathrm{robot}}$ is the current position of the robot, $\boldsymbol{I}$ is the 2×2 identity matrix, and α and σ are parameters to encode, respectively, the constant and distance-dependent measurement noise. For a given sensor measurement, the hypothesis test is repeated for each map landmark in turn. If every hypothesis test rejects the null hypothesis (indicating a mismatch between expected and measured landmarks), then map verification fails, and the path planner does not use map information.

3.2.4 Secure Planner

As a simple abstraction of a vehicle path planner, an A* planner [27] is used to plan coarse paths, which are then given to a lower level path tracking controller.

Predefined *waypoints* along the lane are used sequentially as the planner goal. The externally-provided map includes a *cost overlay*, containing a planning cost for each grid cell in the map. This overlay penalizes travel outside of the lane boundaries with high planning costs. Both the cost overlay and the occupancy grid of nearby obstacles contribute to the complete planning cost function. As the A* algorithm searches for the lowest-cost path, this cost function encodes a preference for lane following and obstacle avoidance in the planner.

The architecture is designed to enforce separation between *trusted* and *untrusted* hardware and software components. As shown in Figure 3.1, the local sensor interfacing (LIDAR, camera, Vicon pose) is performed on a dedicated computing platform, which also receives pose estimates from the motion tracking system communicated over a dedicated wireless network connection. The computing platform is assumed to be a trusted component. Although the component uses wireless communication, in analogous autonomous vehicle architectures where the pose solutions originate from a GNSS-based system, these estimates would be local to the system and hence no network communication would take place. By contrast, the external map server is connected over a separate wireless connection to mimic an external internet connection, and as such is treated as an untrusted component.

A difficulty in enforcing component separation by trust level arises when designing components that use information from both trusted and untrusted sources. In our architecture, both external map information and local sensor data are used for path planning. Any output of a component that uses both these information sources is considered untrusted—the outputs are assigned the lowest trust level of the inputs—however the input data can be explicitly

promoted to a higher trust level, a process known as *endorsement*.

As described in Section 3.2.3, the untrusted map is verified using sensor information. If the map is successfully verified, it is explicitly endorsed for use in the path planning computation. If verification fails, the untrusted map cannot be used in trusted computations, and the path planner falls back to use only (trusted) sensor information to generate a plan. The planning architecture details are shown in Figure 3.1.

The planning and map verification routines are implemented in the *Jif* programming language [36], a Java-like language that enforces information flow control using label annotations to specify security policies regarding confidentiality and integrity. *Jif* ensures these security policies are binding at compile-time and run-time. When the map is verified, run-time language-level endorsement is used to allow the map to be considered by the secure planner. By using a software language that enforces information flow control, we can ensure that untrusted information does not leak into trusted computations before endorsement.

3.2.5 Robotic Platform

A Segway RMP50[2] mobile base is used to actuate the system. The base is statically stable and uses differential drive to maneuver. An existing custom-developed Windows-based control stack is used for path following, based on the pure pursuit algorithm [13]. A 2D SICK[3] LIDAR is used for occupancy grid

[2]Segway RMP, `http://rmp.segway.com`
[3]SICK, `www.sick.com`

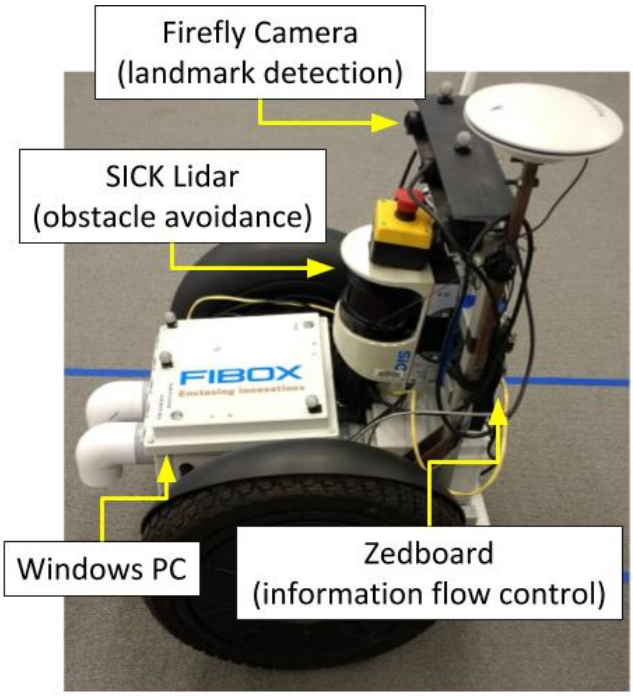


Figure 3.2: Segway robotic testbed.

generation, and a PointGrey[4] Firefly camera for landmark detection. The *Segway PC*, a Windows computer mounted on the mobile base, is responsible for all trusted computation and information handling, including path following, sensor interfacing, and wireless communication with the Vicon[5] motion tracking system. The remote planner is running on a ARM microprocessor, part of a Zedboard[6] assembly.

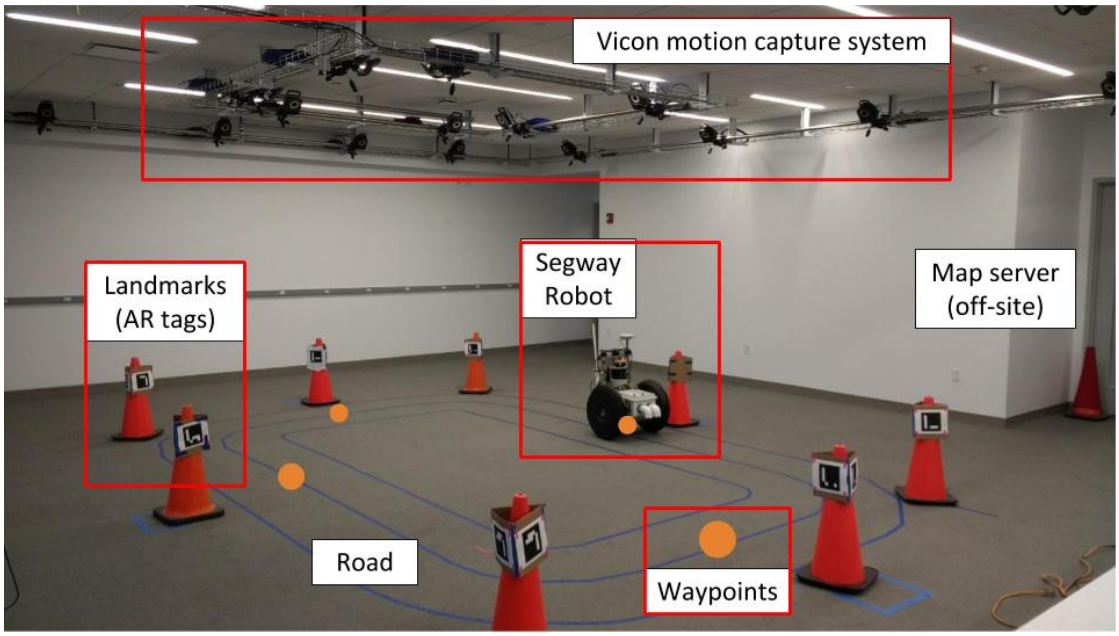


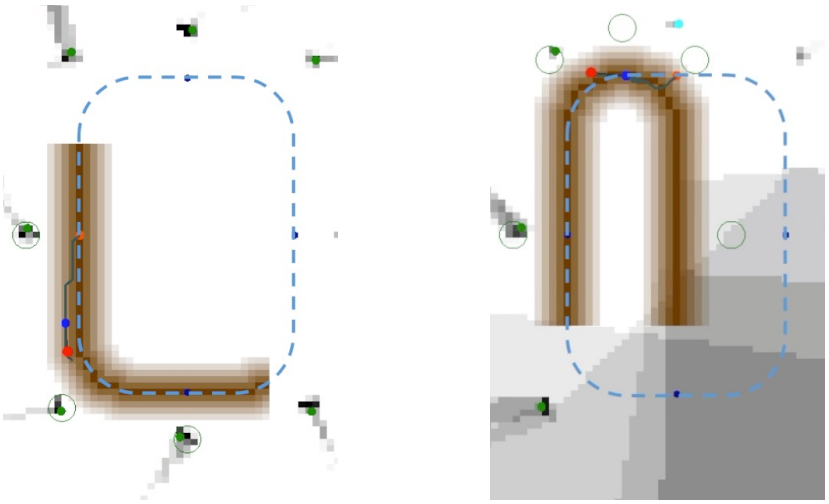
Figure 3.3: Experimental setup. Orange circles represent navigation waypoints provided in software.

3.3 Demonstration Scenario

A demonstration scenario was developed to test the key features of the secure architecture. The scenario is shown in Figure 3.3. A simple loop lane is surrounded by 8 Aruco tag landmarks. Four trusted planning waypoints are provided ahead of time to the control stack, analogous to high-level navigation waypoints from a satellite navigation system or similar. The waypoints are provided to the planner in sequence such that the robot drives around the loop clockwise. The sensor-only contingency plan, in the case of map verification failure, is to decelerate the robot to a stop immediately.

The experiments explored an attack scenario where the external map server

[4]PointGrey, `www.ptgrey.com`
[5]Vicon Motion Systems Ltd., `www.vicon.com`
[6]Zedboard, `www.zedboard.org`



(a) Nominal map.

(b) Malicious map. Note the deformation in the horizontal direction.

Figure 3.4: Segments from the externally provided map. The lane as reported by the map is shown in the brown gradient, and the true lane center is shown in dashed blue. For more detail see the visualization legend in Figure 3.5.

was maliciously compromised and is providing inaccurate maps. The nominal map (Figure 3.4(a)) contains a cost overlay and expected landmark positions which correctly describe the true lane geometry and landmarks. The malicious map (Figure 3.4(b)) is deformed such that the driving loop is narrower than in the nominal case. The landmark locations in the malicious map are similarly transformed, so that they no longer match physical landmarks.

3.4 Results

Three scenarios were tested to evaluate the performance of the proposed architecture. In the first, the nominal case, the map server provides a correct map of lane geometry and landmarks, and the secure planner is used. In the second,

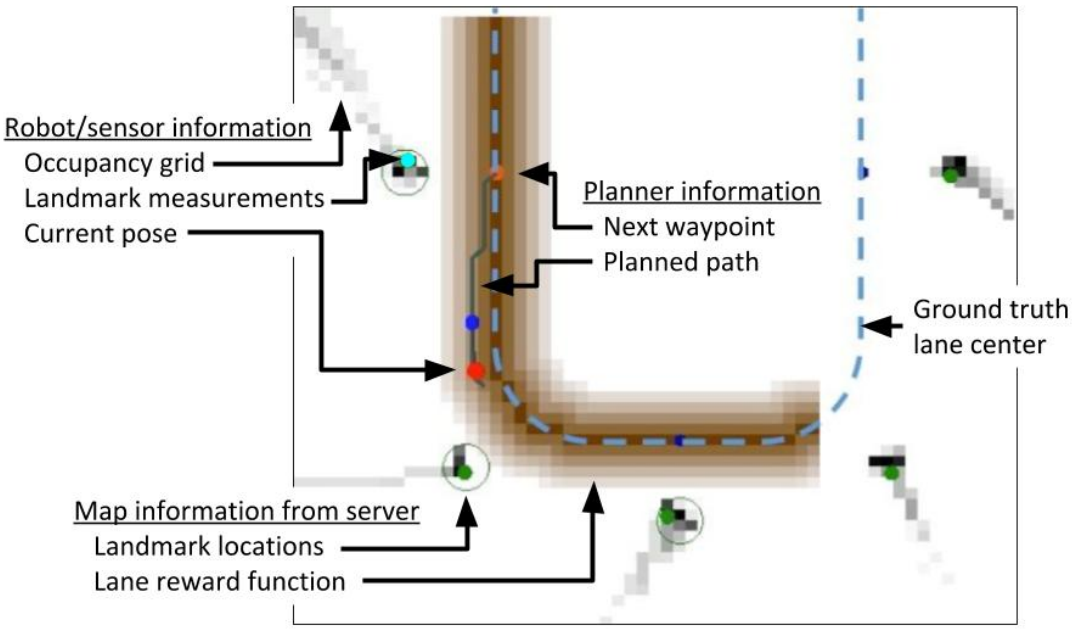


Figure 3.5: Visualization legend.

a malicious map is served to the robot, and a planner is used that does not enforce information flow control. The third scenario involves serving a malicious map to the secure planner. A software visualization (Figure 3.5) was used to present the measurement and map data used in map verification and other decision making. In all scenarios the robot begins at the lower left corner of the track, traveling clockwise.

An image from the first scenario is shown in Figure 3.6. In this test, the robot successfully navigated the loop, as the map was successfully verified due to the agreement between measured and expected landmark locations.

In the second scenario, the planner ignores the trust level of the map information, i.e. it does not check that the map has been endorsed after successful verification. During the initial stages of traversing the loop, the lane geometry of the deformed map is sufficiently similar to the true lane geometry, so the

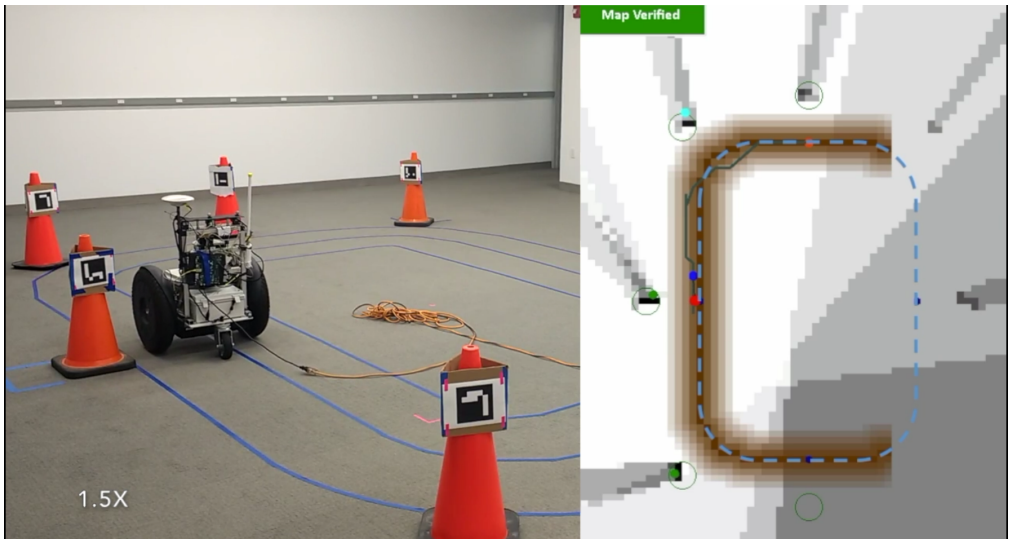


Figure 3.6: Nominal planning operation. The system has verified the map based on the agreement between the measured and expected landmark location, shown in the upper-left corner of the visualization inlay.

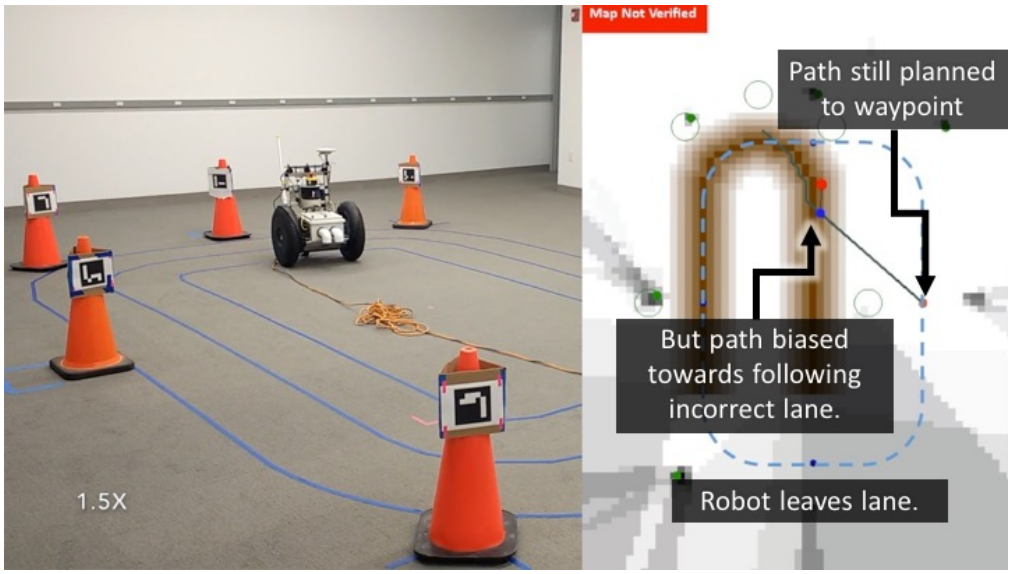


Figure 3.7: A planning failure as a result of the planner ignoring the map verification result. The malicious map lane and corresponding landmarks are deformed relative to the true lane geometry. Despite inconsistent landmark measurements, the map is consumed by the planner, causing the vehicle to leave the lane.

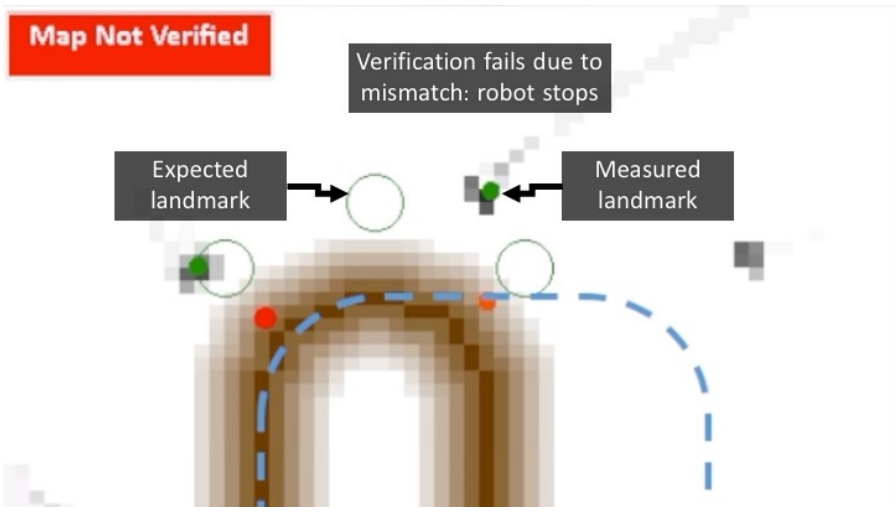


Figure 3.8: A correct planning response resulting from map verification failure.

robot maintains position in the lane. As the robot passes the curve at the top of the map, the lane geometries and landmark locations diverge. Map verification fails, but since the planner does not consider the map trust level, a plan is generated that follows the map-provided lane geometry. This causes the vehicle to leave the lane, shown in Figure 3.7.

The third scenario sees the secure planner served with a malicious map. Again, as even the deformed map landmarks are consistent with sensor observations initially, the map is verified and driving continues nominally. However, as the first curve is driven, the landmark measurements begin to diverge from the expected landmark locations. This causes map verification to fail, and the map is not endorsed for use in the secure planner. The planner therefore generates a sensor-only plan, which causes the robot to come to a full stop before leaving the lane.

Table 3.1: Components in the system and algorithmic architecture of the demonstration platform, compared with analogous components in an autonomous road vehicle architecture.

	Testbed	**Autonomous Car**
Pose	External motion tracking system	GNSS-based pose solution, with map-aided precise localization
Map verification	Visual landmarks	Multi-sensor verification of road features
Sensor suite	Basic vision, 2D LIDAR	Add radar, advanced vision, 3D LIDAR
Planner	Simple A* discrete planner with cost overlay	Dynamics-aware planner with contingencies
Navigation Goals	Hardcoded ahead of time	Generated by navigation software

3.5 Conclusions

We present a secure system architecture and map verification strategy for an autonomous vehicle platform. A proof-of-concept demonstration was developed using an indoor mobile robotic testbed, and experimental results presented. The map verification strategy successfully prevents adverse vehicle behavior in the event of a maliciously compromised map server. By implementing the planning and verification software in the *Jif* language, separation is guaranteed between trusted and untrusted information flows. Additionally, using *Jif* forces the user to consider security when developing software—during the development of the secure planner a number of bugs were automatically discovered where the compiler prevented the use of unverified, untrusted information.

The presented map verification algorithm uses explicit landmarks for verification, which are largely independent of the lane geometry. This process has clear weaknesses: a compromised map could contain correct landmark posi-

tions but an incorrect lane geometry. In a real driving system, verification would be performed with a larger number of measurements sources, using features that are inherently correlated with lane geometry and other relevant driving parameters. For example, lane lines or observed traffic flow direction could be used so that the verification is strongly dependent on information pertinent to driving behavior and is less susceptible to incorrectly specified landmarks. Further extensions of the architecture to a full autonomous vehicle platform are given in Table 3.1.

In an real-world autonomous vehicle platform there are many more components like the path planner, where incoming information flows originate from many different sources with varying trust levels. While in this work we consider a compromised external server in communication with the vehicle, other possible attacks such as sensor spoofing can also be mitigated with a similar methodology, using information flow control and a mechanism for signal verification. A multiple-sensor map verification strategy is essential for mitigating the effects of malicious spoofing of a specific sensor mode, e.g. radar or GNSS.

The secure planner is currently running on a standard ARM processor. The proposed software architecture is designed for eventual implementation on secure processor hardware that is robust to a range of timing channel and other attacks [47]. Secure hardware is especially critical in resource-constrained systems, where it is not feasible to have separate processors for safety-critical and non-critical tasks. In addition to the ARM processor, the Zedboard platform also contains an FPGA for future implementation of a secure processor architecture.

The measurement model used in the proposed map verification algorithm assumes a simple circular Gaussian measurement uncertainty, with covariance

that increases with distance from the robot. A more precise model would consider the error characteristics of the particular sensing modality. Additionally, the map verification process does not consider any negative reasoning about the absence of a particular measurement, it only attempts to align actual sensor measurements with *any* consistent landmark location from the map. More advanced systems would incorporate negative information and occlusion reasoning into the measurement model, to reason about expected measurements that were *not* received.

CHAPTER 4

CONCLUSIONS AND CONTRIBUTIONS

This thesis addresses two challenges that must be overcome to enable safe and secure autonomous vehicles. The first challenge is that faced by autonomous vehicles operating in dynamic urban environments, whereby planning vehicle motion without overly conservative behavior is difficult. The second is of vulnerabilities in common autonomous vehicle architectures that are susceptible to remote attack. As autonomous vehicles become more widespread the potential consequences of a large-scale cybersecurity breach become more severe.

The work in Chapter 2 extends previous work on contingency planning for autonomous vehicles. A metric of *distinguishability* is defined as the probability that discrete modes of an obstacle prediction will be distinguishable at a given point in time. This metric is used to analyze generated obstacle predictions and define the point in time where a contingency is selected for execution: the earliest time where obstacle predictions are sufficiently distinguishable. Simulated experiments show that this planning framework produces safer, less conservative, and lower-cost driving behavior, as compared to existing contingency planners where a contingency is selected at an arbitrary predetermined time independent of predicted obstacle behavior. This improved performance comes at minor additional constant computational cost, and with no change to the computational scaling of the planning framework. Furthermore, clustering of "indistinguishable" predictions, as determined by the proposed metric, provides a principled way of reducing computational requirements. Fundamentally, this work enables contingency planning to be applied to a much wider set of driving scenarios, and is a step toward robust real-world implementation of this

planning concept.

Chapter 3 presents a novel secure architecture for an autonomous vehicle system. The proposed architecture has an emphasis on separation of hardware and software components where possible, to prevent unintentional flows of information from untrusted, insecure components to safety-critical and trusted components. Where software components must communicate for operational reasons, information flow control is enforced at the language level (using the *Jif* language) to prevent untrusted information flows from influencing trusted computation without explicit endorsement. The architecture is prototyped on a mobile robotic platform. To assess security, an external, untrusted map server used to provide simulated malicious data to the robot during testing, analogous to how similar data is served to autonomous road vehicles during operation. A novel map verification algorithm, based on a statistical hypothesis test, is used to ascertain whether the untrusted external data is in agreement with local trusted sensor data. The result of this verification step is used in software to explicitly endorse the map data for use in trusted computations. In an experimental driving scenario, unsafe behavior was able to be generated by using malicious map data without the verification step. With the map verification enabled, the system was able to successfully reject malicious map data by evaluating against local landmark measurements. These experiments represent the first application of the *Jif* language to a mobile robotic system. The architecture is designed for future incorporation of a custom secure multi-core processor that is able to provably enforce non-interference between shared processes.

Both of these pieces of work contribute to important aspects of the road-readiness of autonomous vehicles by improving safety and security of these de-

vices. To the author's knowledge, the work on prediction distinguishability is a novel analysis, and in particular the relationship between perception, anticipation, and planning is one that has not been explored in any depth previously. It seems certain that this relationship must be further developed and better understood before autonomous road vehicles can be considered safe and performant enough for widespread acceptance and adoption.

BIBLIOGRAPHY

[1] Mobileye to generate, share, and utilize vision data for crowdsourced mapping with Nissan. *Mobileye N V*, Apr 2017.

[2] National Highway Traffic Safety Administration. *Vehicle-to-Vehicle Communications*, 2016 (accessed July 29, 2017).

[3] National Highway Traffic Safety Administration. Investigation: Pe16-007, 2017 (accessed July 29, 2017).

[4] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[5] Antoine Bautin, Luis Martinez-Gomez, and Thierry Fraichard. Inevitable collision states: A probabilistic perspective. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4022–4027. IEEE, 2010.

[6] Claus Bendtsen and Ole Stauning. FADBAD, a flexible C++ package for automatic differentiation. Technical report, Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1996.

[7] Binay K. Bhattacharya and Godfried T. Toussaint. An upper bound on the probability of misclassification in terms of matusita's measure of affinity. *Annals of the Institute of Statistical Mathematics*, 34(1):161–165, Dec 1982.

[8] Richard Bishop. *Intelligent vehicle technology and trends*. 2005.

[9] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[10] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

[11] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*. San Francisco, 2011.

[12] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1836–1843. IEEE, 2014.

[13] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon University, 1992.

[14] Alexander G Cunningham, Enric Galceran, Ryan M Eustice, and Edwin Olson. MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1670–1677. IEEE, 2015.

[15] Alex Davies. Google's self-driving car caused its first crash, February 2016. [Online; posted February-2016].

[16] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[17] Dave Ferguson, Michael Darms, Chris Urmson, and Sascha Kolski. Detection, prediction, and avoidance of dynamic obstacles in urban environments. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1149–1154. IEEE, 2008.

[18] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[19] Thierry Fraichard and Hajime Asama. Inevitable collision statesa step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[20] Enric Galceran, Alexander G Cunningham, Ryan M Eustice, and Edwin Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, pages 1–16, 2017.

[21] S. Garrido-Jurado, R. Mu noz Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.

[22] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A survey of

microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, pages 1–27.

[23] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[24] Jason Hardy and Mark Campbell. Clustering obstacle predictions to improve contingency planning for autonomous road vehicles in congested environments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1605–1611. IEEE, 2011.

[25] Jason Hardy and Mark Campbell. Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. *IEEE Transactions on Robotics*, 29(4):913–929, 2013.

[26] Jason Hardy, Frank Havlak, and Mark Campbell. Multiple-step prediction using a two stage gaussian process model. In *American Control Conference (ACC), 2014*, pages 3443–3449. IEEE, 2014.

[27] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[28] Frank Havlak and Mark Campbell. Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments. *IEEE Transactions on Robotics*, 30(2):461–474, 2014.

[29] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. Cross processor cache attacks. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 353–364. ACM, 2016.

[30] Steven G. Johnson. The nlopt nonlinear-optimization package.

[31] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416 – 442, 2015.

[32] Steven M LaValle. *Planning algorithms*. 1999.

[33] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. 2015.

[34] Isaac Miller and Mark Campbell. Rao-blackwellized particle filtering for mapping dynamic environments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3862–3869. IEEE, 2007.

[35] Isaac Miller and Mark Campbell. Particle filtering for map-aided localization in sparse gps environments. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1834–1841. IEEE, 2008.

[36] Andrew C Myers, Lantian Zheng, Steve Zdancewic, Stephen Chong, and Nathaniel Nystrom. Jif: Java information flow. *Software release. Located at http://www.cs.cornell.edu/jif*, 2001.

[37] National Safety Council. NSC motor vehicle fatality estimates, February 2017. [Online; posted February-2017].

[38] World Health Organization. *Global status report on road safety: time for action*. World Health Organization, 2009.

[39] Stephane Petti and Thierry Fraichard. Partial motion planning framework for reactive planning within dynamic environments. In *Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics*, 2005.

[40] Roland Philippsen, Sascha Kolski, Kristijan MaÄ, Roland Yves Siegwart, et al. Path planning, replanning, and execution for autonomous driving in urban and offroad environments. 2007.

[41] Stefan Poslad. *Ubiquitous computing: smart devices, environments and interactions*. John Wiley & Sons, 2011.

[42] Mark L Psiaki. The" blob" filter: Gaussian mixture nonlinear filtering with re-sampling for mixand narrowing. In *Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION*, pages 393–406. IEEE, 2014.

[43] Hugo Reboredo, Francesco Renna, A. Robert Calderbank, and Miguel R. D. Rodrigues. Compressive classification of a mixture of gaussians: Analysis, designs and geometrical interpretation. *CoRR*, abs/1401.6962, 2014.

[44] Andrei Sabelfeld and Andrew C Myers. Language-based information-flow security. *IEEE Journal on selected areas in communications*, 21(1):5–19, 2003.

[45] Jonathan R Schoenberg, Mark Campbell, and Isaac Miller. Posterior repre-

sentation with a multi-modal likelihood using the gaussian sum filter for localization in a known map. *Journal of Field Robotics*, 29(2):240–257, 2012.

[46] Jatinder Singh, Thomas Pasquier, Jean Bacon, Hajoon Ko, and David Eyers. Twenty security considerations for cloud-supported Internet of Things. *IEEE Internet of Things Journal*, 3(3):269–284, 2016.

[47] G Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. Efficient memory integrity verification and encryption for secure processors. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, page 339. IEEE Computer Society, 2003.

[48] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. 2005.

[49] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE, 2000.

[50] Yao Wang and G Edward Suh. Efficient timing channel protection for on-chip networks. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 142–151. IEEE, 2012.

[51] Kevin Wyffels and Mark Campbell. Precision tracking via joint detailed shape estimation of arbitrary extended objects. *IEEE Transactions on Robotics*, 33(2):313–332, 2017.